IndiArt

Project position statement and value proposition

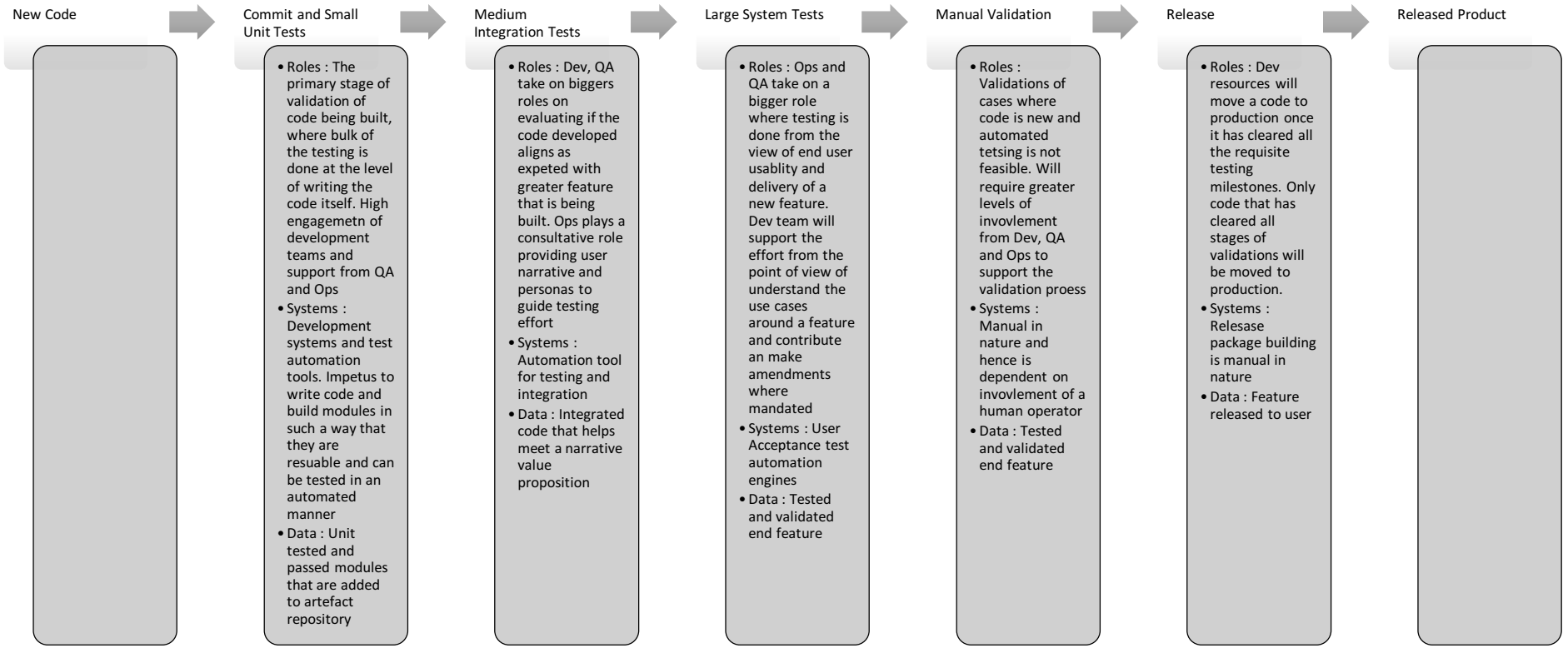
For Home Artists who need a medium to display and market their art, the IndiArt website is a listing and buyer meets seller ecosystem that puts home artists on the same footing as professional artists when it comes to ability to list and sell art. Unlike personal contacts and dependence on individual networks, IndiArt product is global, central, open and doubles as a showcase and marketplace for home artists

Assumption to be tested

If we create a showcase and marketplace for home artists, they will list their art, sell on the platform and continue to engage.

User Story	Test Cases
Create a new entry on the online website for the painting	- IndiArt lets Antara add new art work under her account name - IndiArt asks Antara to feed in descriptive information like description, dimensions, base material, paints used, style etc

Diagram of current process



Whats working well and what needs to be improved

Area	Notes & Questions
Commit & Small Tests	The primary stage of validation of code being built, where bulk of the testing is done at the level of writing the code itself. High engagement of development teams and support from QA and Ops How does this work now? This is done by the developer at the time of coding, there is no second layer of review. The same person writing the code, spends time on performing unit testing How does the commit process work? Once unit level testing is successful the code is committed What happens next? Who does that? The developer then releases the code for Medium integration tests What's our point of view and practices on unit tests? - The unit testing should involve one person doing the coding, and another person reviewing and performing unit level code testing - There should be an effort to ensure that user stories and narratives are shared with the development teams [For each or other discrete topics that come up] Which are the top 3 best things that we do or that we've learned work well for us? Are we doing as much of that as you think we should now? - Unit level coding fixes issues at the stage of writing code, however this can be further enhanced by ensuring developer has complete knowledge of user story and narrative, so then the developer knows why a particular feature is being built What are the top 3 hardest, most annoying things about this process? Which take up a lot of your time? - Prioritizing developer time on bug fixes and UAT testing changes I.e. changes that are identified at the time of medium of large scale testing If you had a whole day free to work on improving this area, what would you do? What if you had a week? What other support or infrastructure would you need to do it? - In order to ensure that code is writing keeping outcome in mind rather than output, developers should be clued in on user stories and narratives
Medium (Integration) Tests	• Dev, QA take on bigger roles on evaluating if the code developed aligns as expected with greater feature that is being built. Ops plays a consultative role providing user narrative and personas to guide testing effort How does this work now? What are the inputs? What are the outputs? What happens next? Unit level code is tested as part of a set of artefacts that contribute to an end feature. Inputs are various instances of unit level code, output is a feature that the be tested against a stated value proposition. Who's involved? How? Dev, QA take a larger role, Ops plays a consultative and provides low/medium engagement on the testing effort t

	[For each or other discrete topics that come up] Which are the top 3 best things that we do or that we've learned work well for us? Are we doing as much of that as you think we should now? Integration of unit level code and time spent on testing at this stage reduces end bugs of a small nature that would waste a lot of time at the time of large scale testing, not to mention the cost implications of re-development and reassign of resource to fix code that is already written What are the top 3 hardest, most annoying things about this process? Which take up a lot of your time? Synchronizing efforts between development and QA teams where there are times where a hand off is required and is not completed successfully, each side claiming that they have done what is expected of them If you had a whole day free to work on improving this area, what would you do? What if you had a week? What other support or infrastructure would you need to do it? Eliminate the requirement of a hand off between completion of development, to hand off to QA and then hand off to Ops. Make the team own the user story, narrative and value proposition collectively
Large (System) Tests	• Ops and QA take on a bigger role where testing is done from the view of end user usability and delivery of a new feature. Dev team will support the effort from the point of view of understand the use cases around a feature and contribute an make amendments where mandated How does this work now? What are the inputs? What are the outputs? What happens next? Inputs are code and feature that has passed the phase of integration testing, output is a feature that is tested against use cases defined at the time of problem definition and identification of value proposition. Manual validations if mandated are then trigger, and if not required the code is deemed fit for release. Who's involved? How? QA and Ops teams play a role as an advocate for the end user narrative, while development team closely supports for bug fixes and changes to original code that are picked up as change requests [For each or other discrete topics that come up] Which are the top 3 best things that we do or that we've learned work well for us? Are we doing as much of that as you think we should now? Measuring the end goal to the original expectation of value proposition, establishing status of feature i.e. which bugs need to be fixed immediately, which can be done post release What are the top 3 hardest, most annoying things about this process? Which take up a lot of your time? Working with Ops expectations that everything will work without any iterations or experimentation around approaches and solutions. If you had a whole day free to work on improving this area, what would you do? What if you had a week? What other support or infrastructure would you need to do it? Build expectations with the operations teams that while the end goal is to provide a fully working end feature, the journey to that point will have some level of iterations and hence requires all stakeholders working together rather than working with an attitude of hand offs.
Manual Validation	How does this work now? What are the inputs? What are the outputs? What happens next? • Validations of cases where code is new and automated testing is not feasible. Will require greater levels of involvement from Dev, QA and Ops to support the validation process

	Who's involved? How? For cases where testing cannot be performed in an automated manner and requires human intervention and QA and the user teams will work closely to manually validate what is delivered against the original goal [For each or other discrete topics that come up] Which are the top 3 best things that we do or that we've learned work well for us? Are we doing as much of that as you think we should now? Manual validation ensures that for features that do not lend themselves to automated testing, the checks and balances exist to ensure that what is delivered is exactly what was expected What are the top 3 hardest, most annoying things about this process? Which take up a lot of your time? The time and effort consumed in manual validations If you had a whole day free to work on improving this area, what would you do? What if you had a week? What other support or infrastructure would you need to do it? Ask the code at the time of being written, also requires that test scenarios, test cases and test scripts are defined in such a way that code can be tested in an automated manner. This will reduce the number of instances where manual validation of code is warranted
Release	How does this work now? What are the inputs? How do we decide when we're ready to release? What's on the checklist? Who checks it? • Dev resources will move a code to production once it has cleared all the requisite testing milestones. Only code that has cleared all stages of validations will be moved to production. Who does the update? The development team will perform the release although this is done manually, and effort is spent on identifying which code is ready for release. How do we make sure it's working? How do we roll back if it's not? Who's involved in assessing and resolving issues? Each code release has a phase where operations team perform a production sanity check on the code, if this is passed the release is allowed to stay. If this fails, the code is rolled back. The sanity check scripts will be the same as final bath of regression tests that were run at the time of large scale feature testing. This testing can be automated so long as regression testing at the stage of Large System Testing was automated in nature [For each or other discrete topics that come up] Which are the top 3 best things that we do or that we've learned work well for us? Are we doing as much of that as you think we should now? Ensuring code in production is released in line with agreements and tweaks done at the time of testing What are the top 3 hardest, most annoying things about this process? Which take up a lot of your time? Identifying which code is ready for release, from a huge maze of various features in various stages of implementation If you had a whole day free to work on improving this area, what would you do? What if you had a week? What other support or infrastructure would you need to do it? Workflow system that will track which pieces of code have passed all mandated validation tests and then release the code to production without requirement manual tracking and list keeping

Assess your highest priority opportunities for improvement.

The following are the areas of development identified in aforementioned analysis

- Automation of code testing, avoid the need for manual validation
- Ensure zero hand off delivery by ensuring the development, testing and operations teams jointly own the feature
- Development team to work focusing on outcome rather than output
- Build acceptance with the operations teams that improvement and feature building is an iterative process and there should not be a commitment from the team to support the process of experimentation and hypothesis testing
- Workflow system that will track which pieces of code have passed all mandated validation tests and then release the code to production without requirement manual tracking and list keeping

While all of these goals are ideal to achieve, the key one in our opinion is “Ensure zero hand off delivery by ensuring the development, testing and operations teams jointly own the feature” because when this is achieved teams stop working in silos and walls do not get built, the other goals then become far easier to get to because the team has already committed to join ownership of the delivery.